

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization  
International Bureau



(43) International Publication Date  
23 October 2003 (23.10.2003)

PCT

(10) International Publication Number

WO 03/088572 A1

Ray

- (51) International Patent Classification<sup>7</sup>: H04L 12/24
- (21) International Application Number: PCT/EP03/03625
- (22) International Filing Date: 8 April 2003 (08.04.2003)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:  
TO2002 A 000325 12 April 2002 (12.04.2002) IT
- (71) Applicant (for all designated States except US): TELECOM ITALIA S.P.A. [IT/IT]; Piazza degli Affari, 2, I-20123 Milano (IT).
- (72) Inventor; and
- (75) Inventor/Applicant (for US only): GHIRARDI, Maurizio [IT/IT]; Telecom Italia S.p.A., Via G. Reiss Romoli, 274, I-10148 Torino (IT).
- (74) Agents: GIANNESI, Pier, Giovanni et al.; Pirelli S.p.A., Viale Sarca, 222, I-20126 Milano (IT).
- (81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MY, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SI, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.
- (84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW),

Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

**Declarations under Rule 4.17:**

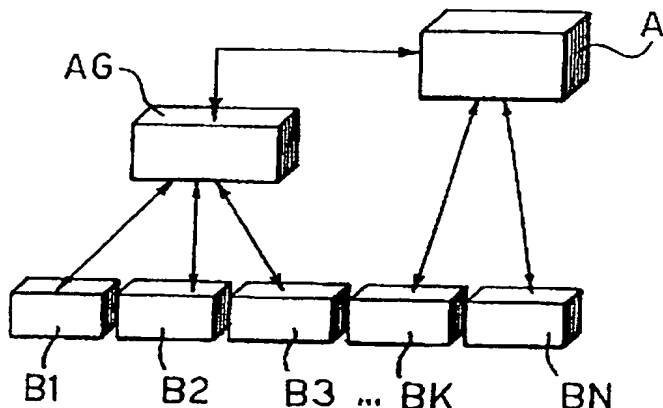
- as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii)) for the following designations AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW, ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG)
- as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii)) for all designations
- of inventorship (Rule 4.17(iv)) for US only

**Published:**

- with international search report before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments

[Continued on next page]

(54) Title: METHOD FOR ORGANISING COMMUNICATION BETWEEN MANAGER OBJECTS AND MANAGED OBJECTS IN A COMMUNICATION NETWORK, ARCHITECTURE AND SOFTWARE THEREOF



(57) Abstract: A method for managing at least one managed object (B1,..., BN) through a communication network (R) by at least one manager object, comprising the following operations: - providing at least one intermediate object or hierarchic agent (AG) configured to manage said at least one managed object (B1,..., BN) according to a data set (1100), said management being translated into a set of results (1104), - providing said data set (1100) from said at least one manager object (A) to said intermediate object (AG), - managing said at least one managed object (B1,..., BN) through said at least one intermediate object (AG), to generate said set of results (1104), and - transferring (1108) said set of results (1104) from said at least one intermediate object (AG) to said at least one manager object (A). Preferably, communication between manager object (A) and intermediate object (AG) implements UDP protocol and compressed

WO 03/088572 A1

mode.

WO 03/088572 A1



*For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*

**"METHOD FOR ORGANISING COMMUNICATION BETWEEN MANAGER  
OBJECTS AND MANAGED OBJECTS IN A COMMUNICATION NETWORK,  
ARCHITECTURE AND SOFTWARE THEREOF"**

**Technical Field**

This invention refers to the methods for establishing a communication between at least one manager object (hereinafter called "manager" in brief) and at least one managed object (hereinafter called "agent" in brief) in the context of a telecommunication network.

**Background Art**

A typical reference architecture used for this purpose is shown in figure 1 illustrating the connection between a manager module A and a certain number of agent elements B1, B2, B3, ... interconnected by a telecommunication network R.

This architecture, for example, is described in SNMP (Simple Network Management Protocol) specifications. Reference can be made to document RFC1157, revision 1990.

On a general level, Internet protocol architecture implements four logic levels, commonly called Application (A), Transport (T), Network (N) and Link (L).

As shown in figure 2, each level is in fact nested within the protocols underneath. For example, Application level protocols, such as the previously mentioned SNMP and TFTP, i.e. Telnet or FTP protocols, are nested within the protocols underneath.

Specifically, SNMP and TFTP protocols are nested in UDP (User Datagram Protocol), which in turn is nested in IP (Internet Protocol), and consequently injected in the physical carrier (cable, fibre, radio waves), indicated by reference D, by software drivers or hardware devices, to implement the Link L on a physical level.

Similarly, Telnet and TFTP protocols are nested in TCP (Transmission Control Protocol) which in turn is nested in

the IP protocol and consequently injected in the physical carrier device L.

The main characteristics of TCP and UDP on the transport level can be described in the following terms.

5 TCP is a system communication oriented protocol (systems are identified by a network address) and it is linked to the software it employs. A connection, i.e. a permanent communication with the remote system, must be established before establishing a communication using this  
10 protocol. Data transfer is controlled and guaranteed but slow, especially when discontinuous or small. The delay is caused by the characteristics of the IP protocol and by the fact the connection is created after each request and removed, i.e. shut down, at the end of the communication if  
15 it is not used. The communication is costly in terms of employed system resources due to the complexity of the protocol and the checks to which data and connection are subjected to ensure correctness of the communication.

Conversely, the UDP protocol is process communication  
20 oriented and process communications are identified by logical ports each characterised by a number in the range from 0 to 65535. For transmission, the protocol accepts messages from various application procedures and passes them to IP protocols for transmission. This function is  
25 called multiplexing. For reception, the UDP protocol receives data packages from the IP layer via a destination application process. This function is called demultiplexing.

The UDP protocol is much lighter in terms of resource  
30 utilisation and is simple to implement and manage. Specifically, it employs a brief sixty-four bit header (called "PDU User Header") divided into "source port", "destination port", "length" and "check sum" and a ninety-two bit header containing the "source address",

"destination address", "filler", "protocol type", "length PDU" fields.

The UDP protocol is fast because the IP transmission protocol does not require processing or checks; it simply  
5 transmits, where possible, from the current network address to the destination network address.

The native UDP protocol does not employ reception acknowledgement messages, does not sort messages, does not check flow and consequently is not completely safe or  
10 reliable because messages may be lost, rejected, duplicated, received in the wrong sequence or the arrival rate may be higher than that of the application and receiving process in the network.

A generic process architecture employing UDP as  
15 transmission protocol is characterised by being associated to a port according to criteria schematically illustrated in figure 3.

Two basic criteria are used to assign numbers to the reception and transmission ports.

20 A first criterion consists in defining universal assignments in which the respective numbers are official defined and recognised by all parties.

A second criterion consists in defining dynamic binding according to which a program asks for a port whenever  
25 needed and the port is assigned by the network software. Reception ports are normally pre-assigned even if they may be modified. Transmission ports may be defined by using either of the two methods.

Specifically, in the diagram shown in figure 3,  
30 reference PA generically indicates the various application processes (1, 2, 3 ...) which interface with the UDP protocol via three respective ports.

Additional integrative components called "physical objects" and "logical objects" can be identified, in

addition to application process in UDP protocol based management architectures, as shown in greater detail in the diagram shown in figure 4.

Application processes can be additionally split into 5 originator application processes with manager function, generically indicated by reference A and one (or more) destination application processes (called agents) according to the role performed at the time.

The term "physical object" is used for hardware 10 containers or media (e.g. a personal computer) which host other physical objects needed for application operation. The containers are identified in the diagram shown in figure 4 by references  $P_1$  and  $P_2$ . Additional physical objects include the physical (e.g. a RAM) and/or virtual 15 (e.g. file) processing memory and the central processing unit (CPU) employed by the hardware medium or media for running the processes (software, basic firmware, protocols, applications). Memories of this sort are shown in the diagram in figure 4 by references  $R_1$  and  $R_2$ .

20 In the diagram shown in figure 4, reference W indicates the system software on operating system level and references Y and Z indicate the software consisting of one or more application software oriented protocols (transport) and the network board or CR transducer oriented software 25 (in this case consisting of one or more protocols) for interfacing with the network R.

The following description refers to the diagram in figure 4, which shows the relations between application processes, objects and architectural components. A system 30 software W is used to perform the assigned tasks in a system or device employing an available portion of the processing memory  $R_1$ ,  $R_2$ .

A process A residing in system  $P_1$  interfaces with a process B residing in system  $P_2$ , through components W, Y

and Z necessarily present in both devices, network boards and physical medium.

The software components A, B, Y, Z and W use and share a certain amount of memory  $R_1$ ,  $R_2$  according to their 5 characteristics.

The maximum usable band is linked to the characteristics of the network R and of the network boards CR, which must necessarily coincide.

The possibility of using an architecture of the type 10 shown in figure 1 is conditioned by a number of factors.

Firstly, the maximum usable band of the network R is conditioned by the number of managers and agents and the amount of generated traffic. The usable band may be maximum only when there are only two devices, i.e. one manager and 15 one agent. The usable band must be shared if there is one manager and several agents. Consequently, the maximum usable band for each communication between manager and agent cannot be guaranteed.

Generally, the communication between a manager and 20 several agents can be managed either by means of a sequential strategy or a parallel strategy.

In the sequential strategy, the manager establishes a communication with an agent and waits for the communication to end before continuing with the next communications.

25 The parallel strategy exploits the multiplexing and demultiplexing functions offered by the protocol (which is typically a UDP, User Datagram Protocol) through a competitive dynamic port assignment mechanism to establish a certain number of simultaneous communications with 30 several agents.

According to the sequential method, both the manager-to-agent output band (i.e. the sum of total transmitted message sizes divided by the time employed to send them) and the manager input band (i.e. the sum of the total

message sizes received by each node manager divided by the time required by the manager to receive them, the reception time being the sum of the agent processing time plus the network delay) are low, because transmission and reception times are long.

According to the parallel method, the manager-to-agent output band is high and the input band may be very high, because transmission and reception times are very short and because the replies are certainly greater than required.

10 The architecture according to the known art shown in figure 1 presents many limitations and shortcomings.

Sequence transmission is not effective when the number of agents exceeds a certain value (e.g. one thousand). This is because the time required to complete activities considerably increases. Furthermore, the architecture in figure 1 generates traffic bursts, also of large dimensions, due to the fact that the traffic generated by the simultaneous requests by managers and the return traffic generated by the agents may occur at the same time.

20 This may exceed the available band limits with consequently degraded network functionality and loss of messages.

The parallel method employs several manager processes assigned to different UDP ports and this may use up all system resources, such as RAM and CPU.

25 The protocols used by application process, e.g. the aforesaid SNMP protocol or the TFTP Trivia File Transfer Protocol (see document RFC1350), are not optimised for transporting large amounts of information or for working in networks with a high number of protocols. Additionally, the protocols are of the point-to-point type and consequently

30 multilevel architectures cannot be implemented and managed.

Furthermore, as shown in the architecture illustrated in figure 1, all agents should in some way be directly reachable by the manager. The agents which cannot be



directly reached by the manager, e.g. because they are connected to different networks from the manager, require the installation of a dedicated manger in order to be managed.

## 5 Disclosure of the Invention

The object of the invention is to provide a solution capable of overcoming the aforesaid shortcomings.

According to the invention, this object is obtained by a method whose characteristics are specifically recited in  
10 the annexed claims. The invention also relates to the corresponding network architecture and the corresponding software, i.e. the software which can be directly uploaded to the memory of a digital processing unit comprising software code portions capable of implementing the method  
15 according to the invention when the software is run by at least one digital processing unit.

Essentially, the solution according to the invention implements an optimised multilevel management architecture to subdivide management activities over several machines  
20 whereby overcoming the limitation related to the need of utilising a traditional single level architecture. All this limits the use of the band, specifically as concerns the possibility of optimising utilisation of physical manager resources.

25 Briefly, the solution according to the invention consists in realising an intermediate object, i.e. a new type of agent, called "hierarchic agent" capable of receiving sufficient information from the manager to perform the management activities that the manager would  
30 perform directly on the controlled agents.

Consequently, as better described in the description that follows, the solution according to the invention is suitable and particularly advantageous when used in combination with compressed UDP message transfer methods.

**Brief Description of Drawings**

The invention will now be described, by way of example only, with reference to the accompanying drawings wherein:

- figures 1 to 4 refer to the known art and have been  
5 described above,
- figure 5 illustrates the new management architecture according to the invention in general terms,
- figure 6 illustrates a first form of embodiment of the solution according to the invention,
- 10 - figure 7 illustrates a second form of embodiment of the solution according to the invention,
- figure 8 shows an operative logic example of an architecture according to the invention,
- figure 9 illustrates a possible communication management  
15 diagram within an architecture according to the invention,
- figure 10 illustrates the shared agent management method,
- figure 11 illustrates the architecture of a so-called hierarchic agent according to the invention,
- 20 - figure 12 illustrates a possible organisation of the structure and nesting of controls supported by a hierarchic agent within the scope of the invention,
- figures 13 to 15, each split into two parts referred to transmission (part a) and to reception (part b),  
25 illustrate some preferred forms of embodiment of the solution according to the invention in the form of flow charts,
- figure 16 is an additional flow chart illustrating the more general characteristics of the solution according to  
30 the invention, and
- figures 17 and 18 illustrate additional possible embodiments of the solution according to two possible variants of the invention.

**Best mode for Carrying Out the Invention**

The diagram in figure 5 shows the general architecture according to the invention.

By direct comparison with the diagram in figure 1, an additional module, i.e. an intermediate object called  
5 hierarchic agent AG, is integrated in the architecture according to the invention based on the presence of a manager A and a plurality of agents B1, B2, B3 which reciprocally interface.

Essentially, the hierarchic agent AG interfaces with  
10 the manager A so as to receive a sufficient amount of information from the manager A to carry out the same management activities of manager A on a certain number of agents B1, B2, B3 (any number of agents is possible).

In the solution according to the invention, manager A  
15 may continue to preserve and directly control other agents, indicated by references Bk, ..., BN in the diagram in fig. 5.

Obviously, any number of the number of agents B1, ..., BN and any number of divisions for management purposes  
20 between hierarchic agent AG and manager A can be implemented.

The architecture schematically shown in figure 5 is used to create multilevel architectures without duplicating manager functions because a new element (i.e. the  
25 hierarchic agent AG) may be used to carry out the necessary activities and obtain the required results.

In practice, the intermediate objects called hierarchic agent AG is capable of receiving suitable formatted messages from manager A by being presented as an agent (B1,  
30 B2, B3, ...), e.g. SNMP messages containing sufficient information to perform the activities required by the manager A on specific agents identified by means of a network address using specific protocols (SNMP, TFTP, Telnet, DNS, etc.). After the required activities, the AG

module sends the results to the manager A. The interconnection between manager A and hierarchic agent AG may be implemented using the network R (as in the form of embodiment shown in figure 6) or using different networks via a double connection (as the form of embodiment shown in figure 7, where references RP and RA indicate the two networks).

The diagram in figure 7 shows the extreme flexibility of the solution according to the invention.

Specifically, the diagram shows that the first network, indicated by reference RP, may be used for communications between manager A and an agent B1 which continues to be managed directly by the A, for allowing communications between manager A and hierarchic agent AG which "replaces" the manager A in the management of the agents B2 and B3 in a second network indicated by reference RA.

The solution according to the invention also optimises use of the network (or of the networks, in general) in terms of band.

Specifically, algorithmic compression is preferably applied on the data contained in the application protocols (OID in SNMP, payload in UDP, etc.) to reduce network traffic due to communication between manager A and hierarchic agent AG.

This method essentially consists in subjecting (at least) the message payload to a compression operation preferably based on acknowledgement of sequences which periodically appear in the message. In a specifically preferred way, this compression operation is carried out according to a gzip method, such as zLib.

This method, which will be seen in greater detail with reference to figures 12 and following, reduces the number of exchanged PDU data packages in favour of higher data contents.

For example a data PDU in the TFTP protocol uses 516 bytes and consequently 1016 packages are required to transfer a 520 Kb file. Following compression according to the aforesaid criteria, 520 Kb is reduced to 4 Kb, which means that they can be carried in a single UDP data package or SNMP message. Transfer consequently consists in transporting "equivalent application messages" instead of "single messages". This means that the quantity of generated traffic can be reduced, transferred data being equal.

The illustrated solution can also be used to optimise system resources needed to perform the activity.

This is because the illustrated solution distributes the manager's activities over several hierarchic agents AG, according to the criteria shown in figure 10, for example. In this figure, references AG1 and AG2 indicate two hierarchic agents which co-operate with a manager A to manage a certain number of agents B1 to B5, the management of one or more agents (agent B3, in the example shown) being shared by two hierarchic agents AG1 and AG2.

The possibility of sharing the activities of manager A over several hierarchic agents may be exploited to use the resources (CPU, RAM, etc.) which are free at the time. Return traffic to the manager A - generated only at the end of the activities - exclusively by hierarchic agents AG1 and AG2 consists in the results transferred by the hierarchic agents AG1 and AG2 to the manager A. This occurs preferably according to the compressed method, consequently by employing a few packages whose size is medium-to-small and whose data contents is high. These packages do not affect system resources of the manager A needed for decompression and management.

In this way, as shown in the diagram in figure 8, the interaction between manager A and hierarchic agent AG

(reference will be made to a single module on this type in the text that follows for the sake of simplicity and extension to several hierarchic agent modules will be obviously understood) is based on performance of micro-  
5 activities and exchange of signals useful for management.

Specifically, in the step indicated by reference 1100, manager A sends an activity request to hierarchic agent AG in the form of messages, e.g. by using a standard or compressed SNMP message. In the step indicated by reference  
10 1102, the hierarchic agent AG receives and analyses the request, starts processing and collecting information. During processing - step 1104 - hierarchic agent AG sends statistic messages and messages for synchronising the status of activities in progress to the manager: this  
15 occurs in the step indicated by reference 1106. Having ended the management activities of the managed objects, i.e. the agents, the hierarchic agent AG sends the results to the manager A. This occurs in the step indicated by reference 1108. In the step indicated by reference 1110,  
20 the manager A receives and processes the results of the activity by sending a result reception acknowledgement message to the hierarchic agent in step 1112. The hierarchic agent AG then ends the required activities in a step indicated by reference 1114.

25 This cycle may be repeated several times by the manager according to the obtained result. For example, new requests may be sent if some data are considered not sufficient.

The diagram in figure 9 describes the high level logical elements of the hierarchic agent AG and the  
30 relations with other components of the architecture.

It will be noted that the diagram in figure 9 essentially corresponds to the architecture in figure 5, in which manager A directly manages some agents Bk, ..., BN, and delegates the management of other agents B1, B2 and B3

to hierarchic agent AG. The interfacing of the network R is shown in figure 6.

The manager A interfaces with its direct agents and with the hierarchic agent AG through communication 5 implementing UDP protocol, e.g. using standard or (preferably) compressed SNMP messages.

For this purpose, in addition to the control and management logic LCG, the hierarchic agent includes two modules (indicated by references ARX and ATX, respectively) 10 for managing communications between the hierarchic agent AG and the manager A so that the hierarchic agent AG may be "seen" by the manager A essentially as if it were another agent managed directly by the manager A.

The hierarchic agent AG additionally comprises a multi- 15 manager module MM which superintends communications between the hierarchic agent AG and the agents B1, B2, B3, so that each agent may essentially "see" the hierarchic agent AG as if it were the manager A.

The manager A and the multi-manager component of the 20 hierarchic agent AG communicate with the various agents using standard methods/protocols.

Consequently, as shown in the diagram in figure 10, one agent (agent B3 in the example shown) may be managed by one or more hierarchic agents AG1 or AG2 controlled by the same 25 manager.

The diagram in figure 11 illustrates the internal architecture of the hierarchic agent AG for implementing the result management and control logic.

In a preferred way, the solution according to the 30 invention is based on complete message compression (header, indicated by references MH, and PDU).

Specifically, two possible different transfer methods or embodiments are employed.

The first nests the SNMP message in a new compressed SNMP

message and sends it using standard UDP.

The second controls the UDP directly via a driver providing the result of the SNMP message compression as Data Octet.

The compression method is essentially based on the  
5 acknowledgement of sequences which appear periodically in the message.

In a particularly preferred form of embodiment, a variant of the method known as LZ77 is employed as compression method (see Ziv. J., Lempel A., "A Universal  
10 Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, Vol. 23, No. 3, p. 337-343); the method is well known in UNIX environment. It is called gzip (gzip format - RFC 1952) and used also by the popular PKZIP application. The specifications of this  
15 method are of public domain. Source libraries are available for implementing and using these solutions in various development environments and operating systems, such as HP-UX, Digital, BeOS, Linux, OS/2, Java, Win32, WinCE.

Specifically, algorithm porting can be used on win32  
20 using the "zLib" library. The main characteristic of the library is to allow runtime and on-memory compression of both binary data structures and strings, which is a fundamental factor in system performance.

The diagram in figure 11 shows the ARX modules and ATX  
25 mentioned above with reference to figure 9.

The ARX module is exclusively responsible for collecting the messages from the network R passing them to an input queue I included in a queue management module indicated by reference G.

30 Symmetrically, the ATX module is exclusively responsible for sending the messages from the output queue of the queue manager G indicated by reference U.

Queues manager or Module G is exclusively responsible for analysing the messages from the input queue I, the



output queue U and another queue (called working queue) indicated by reference L at each clock pulse.

Said clock signal is generated by a timer module or timer indicated by reference T, which is exclusively  
5 responsible for generating the synchronisation clock of the queues manager G.

The messages in the input queue I are taken and sent to a DC module, which is an interpreting module (and decompression module, in a preferred embodiment) of the  
10 messages, for future processing at each clock signal generated by the timer T. Said decompressing/interpreting module is indicated by reference DC.

Furthermore, the messages in the working queue L are analysed at each clock signal generated by the timer T; a  
15 message indicating activity status in the output queue U are generated for each message in the queue.

At each clock signal from the timer T, the messages in the output queue U are transmitted to the manager through the ATX module.

20 In greater detail, the DC module is exclusively responsible for analysing each message received by the input queue I, decompressing it if required and sending it according to priority to an activity co-ordinating module CA indicating the method and type of activity to the  
25 performed.

The CA module essentially is responsible for:

- instantiating a concurrent process suitable to the request of the message interpreter by co-ordinating activities through a manager controlling module, indicated  
30 by CM, and monitoring the status;

- updating activity status of the requests in the working queue L, and

- creating statistic check messages to be sent to manager A through the output queue U, these messages

containing statistic information on the overall operation status of the instancing concurrent processes.

The CM module is responsible for managing both in a co-ordinated and separate way possible other protocol manager  
5 modules (of which three are shown herein by the way of example, indicated by references MP1, MP2, MP3) by collecting and analysing the received information. At the end of the activity, the result is sent to a message compiling and compressing module, indicated by reference  
10 CCM, in view of the subsequent insertion in the output queue U for subsequent transmission to the manager A.

The CCM module is exclusively responsible for managing the result of the activities generated by the concurrent process, creating the message and possibly compressing it  
15 if the request was compressed, placing it in the queue manager output queue.

Each of the modules called protocol managers MP1, MP2, MP3 (as mentioned there can be any number of managers according to the number of protocols to be managed) is  
20 responsible for communicating with agents through a specific respective protocol (e.g. Telnet, SNMP, TFTP, etc.).

Within the system, a univocal identification number is assigned to each message for rapid, error-free  
25 identification in the architecture. The characteristics of the hierarchic agent AG agent architecture can be outlined as follows.

The hierarchic agent AG is configured as a lighter module with respect to a traditional manager because simple  
30 components are used to emulated network protocols (e.g. SNMP, Telnet, DNS, TFTP, etc.). For example, in the case of the TFTP protocol, only the basic units which make communication with agents compliant with the protocol are implemented instead of the entire server.

The hierarchic agent AG is also faster than a traditional manager because it uses and optimises only the RAM of the host system without accessing disks or databases, for example, which is notoriously slower. 5 Additionally, the hierarchic agent AG does not contain complex manager type message processing functions and is consequently more effective with respect to a traditional manager in terms of resource use being activated only by reception of a request from manager A and deactivated at 10 the end of the activity.

The described architecture is used to implement several simultaneous, co-ordinated activities which involve several protocol typologies, providing the possibility of accessing the agents in hierarchic mode, i.e. allowing that a certain 15 agent may be reached by two hierarchic agents, the first of which acts as a primary element and the second which acts as a secondary element to the manager A.

This means that the secondary hierarchic agent can be used when the first is not available.

20 The described solution is consequently more robust in the event of failures or, in general, (also temporary) unavailability of certain elements.

The availability of ARX and ATX modules for disjointed two-way communication means that high volumes of input 25 traffic can be managed without compromising transmission performance. Particularly, ATX module transmission is managed according to a method which may be called either "timed" or "gaussian" for blocks of messages according to the respective priorities. This approach avoids possible 30 bursts of traffic because a predetermined number of messages is sent at each clock signal according to priority (e.g. twenty priority "1" messages, ten priority "2" messages, eight priority "3" messages, two priority "4" message and one priority "5" messages), while the remaining

messages are queued and sent during the next cycle. Additionally, this prevents the formation of "bottlenecks" because each message flows from one module to the other through internal buffers which disjoin the module  
5 processing speeds.

In a particularly preferred embodiment, the structure of the messages used for communication between manager A and hierarchic agent AG, according to the method illustrated in figure 12, presents a header I followed by a  
10 data body CI.

In this specific case, the header I typically contains the following information:

- message format version (e.g. 1.0),
- maximum command processing time (in milliseconds),
- 15 - compressed contents indicator (1 = encoded, 0 = non-encoded),
- error description (compiled in messages confirming the absence of errors or containing the text of the error),
- message size in bytes,
- 20 - IP address of the agent on which to perform the activity,
- priority of the message indicated by the manager (0 = priority assigned by the hierarchic agent AG, 1 = maximum, 5 = minimum),
- 25 - identification of the manager of protocol to be used,
- required activity typology (the command itself),
- source UDP port of manager request,
- version of manager A or hierarchic agent AG,
- univocal identification of request within manager.

30 The data body CI contains specific information for the protocol manager (MP1, MP2, MP3, ...) to be used to perform the required activities. These indications differentiate according to the activities to be performed and the protocol used. The following contents may be expressed, for

example, as follows:

- SNMP procedure: contains a standard SNMP message with  
OID SNMP to be requested, typology of operation to be  
performed (GET, GET NEXT, SET and BULK, etc..),
- 5     - Telnet procedure: contains authentication parameters  
(UID, password), operator command, indication whether to  
return outputs generated by commands,
- SNMP procedure: contains OID SNMP of all MIB branches  
to be collected through the standard SNMP operation
- 10    typology (BULK or GET NEXT),
- co-ordination procedure: contains multi-protocol co-  
ordination method in form of script,
- TFTP file management procedure (non standard):  
contains activity typology (upload or download), list of
- 15    files to be collected or downloaded,
- agent reachability test procedure: contains test  
typology/typologies to be performed via DNS look-up and  
reverse DNS look-up, ping, Telnet and SNMP port  
reachability,
- 20    - hierarchic agent reachability test procedure: does  
not contain activities to be performed and is used by  
manager A to test reachability of the hierarchic agent AG,  
and
- statistic transmission command: contains data for
- 25    registering and defining the UDP port of manager A to which  
statistic data must be sent.

Commands in turn may be compressed and nested using an  
algorithmic compression method consisting of a compression  
operation, specifically based on acknowledgement of

30   sequences which appear periodically in the message.

Specifically, as shown in the diagram in figure 12, the  
header I and the data body CI of the message may be nested  
in a message structure supported by the hierarchic agent AG  
comprising a message header MH and the remaining part PDU

to generate a SNMP or UDP message susceptible of transiting on IP level.

The flow charts in figure 13 illustrate the method used for compressing (figure 13a) and decompressing (figure 13b) the SNMP message.

The flow charts in figure 14 shown (again with reference to figure 14a for transmission and to 14b for reception) a first solution in which a compressed SNMP message is transferred via SNMP nesting.

10 The flow charts in figure 15 refer to a transfer solution via UDP nesting. Separate reference is again made to transmission (figure 15a) and to reception (figure 15b).

The diagrams in figures 17 and 18 refer to the total compression and transmission operations exemplified in part 15 of a) of figures 13 and 14 (figure 17) and part a) figures 13 and 15 (figure 18), respectively.

In the flow chart in figure 13, reference 100 indicates the step in which the entire SNMP message (header + PDU) is read and converted, in a subsequent step indicated by 20 reference 102 into hexadecimal format. This occurs by applying BER type encode.

The message encoded in this way is compressed on-memory using a compression method based on the acknowledgement of a recurrent sequence, such as the method documented in the 25 previous mentioned zLib library.

This occurs in a step indicated by reference 104 to obtain a compressed data unit which is ready for transmission in step 106.

Symmetrically, the flow chart in part b of figure 13 30 comprises four steps 206, 204, 202 and 200 (intended to be processed in the order shown), in which the received compressed data unit (step 206) is subjected to decompression (step 204) in view of subsequent hexadecimal decoding (step 202) with subsequent reconstruction of the

internal SNMP message (step 200).

The numeral references of the steps in the flow chart in part b) of figure 13 are reversed with respect to the processing order simply to highlight the symmetry with 5 steps from 100 to 106 of the compression procedure. Similar choices are made with reference to the flow charts in figure 14 and figure 15.

As indicated, figure 14 and figure 17 refer to a transfer solution in which a compressed data unit is nested 10 in a standard SNMP message characterised by a variable binding and standard UDP transmission method.

The compressed data unit nesting method in step 106 consists of an initial step (indicated by reference 108) in which the compressed data unit is read in bytes and then 15 transposed (in a subsequent encoding step indicated by reference 110) into the corresponding set of ASCII characters.

The variable binding of the message consists of a first numbered OID (e.g. 1.3.6.1.4.666.1) which contains the 20 value of the \_ZIP\_xxxx string (where xxxx is the size of the original file) and is generated in the subsequent step, indicated by reference 112 (possibly after auxiliary functions such as ACK TAB + NULL - see block 110a in figure 17). Proprietor code 666.1, which is currently not 25 registered by IANA Internet Assigned Numbers Authority, has been used in the example above.

The subsequent variable binding elements which contain the compressed data unit translated into ASCII consist of OID/value pairs. The value contains portions of the 30 compressed data unit transposed into ASCII format whose maximum size is 255 characters.

The header data of the SNMP message are then reconstructed. This occurs in step 112, which is followed by step 114 in which additional encoding according to BER

method is performed to generate the PDU UDP payload intended to be used for sending data (step 116).

Also in this case, the steps indicated by references 216, 214, 212, 210 and 208 in part b) of figure 14, intended to be processed in the order in which they were shown above, are dual functions intended to be implemented during reception of steps from 108 to 116 referred to transmission.

By adopting the solution illustrated in figures 14 and 17, the compressed SNMP message has a standard SNMP logic format and a proprietor content. Functional extension (minimal) is consequently required by the agent's manager to permit acknowledgement and encoding/decoding.

Experiments conducted by the Applicant demonstrate that this solution is entirely feasible without negatively affecting network architecture.

The alternative solution (to which reference is made in figures 15 and 18) consists in preparing the compressed data unit starting from the SNMP message according to the method illustrated in figure 13 followed by direct nesting of said data unit in the PDU UDP payload.

Naturally, to ensure correct operation, this solution requires availability of a dedicated transmitter and receiver (e.g. such as the ARX and ATX modules in figures 9 and 11), for example in conditions requiring the availability of a different UDP port than standard. The transmitter must consequently know which UDP port is used by the receiver and vice versa. Information on used ports can be exchanged on higher level by means of a synchronising message in standard SNMP format according to the criteria which are described in greater detail below.

By adopting the alternative solution illustrated in figures 15 and 18, the compressed data unit made available during step 108 and intended to replace the BER in the



message becomes the payload of the UDP message.

The respective operation is outlined in the step indicated by reference numeral 120 in figures 15 and 18. This step preceeds the transmission step 122 destined to the respective dedicated port (called port X in general) of the receiver.

Also in this case, the complementary operation consists of three steps indicated by references 222 (reception via port Y of the module which is receiving at the time), 220 (PDU UDP payload extraction) and 218 (creation of compressed data unit intended to be transferred to step 206 in the flow chart in part b of figure 13).

Also in this case, steps 222, 220 and 218 are performed in the order in which they are shown.

The synchronisation message to which reference was made in the description above is sent by the manager A to the hierarchic agent AG according to a general application-to-application criteria using standard SNMP format containing a proprietor variable binding.

Transferred data types are:

OID	Value
1.3.6.1.4.666.2	<UDP_TX_Port>
1.3.6.1.4.666.3	<UDP_RX_Port>

Manager A sends a proprietor message to hierarchic agent AG compiling <UDP\_TX\_Port> with the number of the port intended to be used for UDP transmission (e.g. 1024) and <UDP\_RX\_Port> with the number of the port used for UDP reception (e.g. 1224). The hierarchic agent AG replies to manager A by sending a similar message containing its own information. This method reduces processing time by improving solution efficiency.

The flow chart in figure 16 additionally shows how the described solution may be generalised and applied to any type of message employing UDP for transport (e.g. SNMP,

PING, etc.). This generalisation can be exploited to create a UDP driver capable of replacing those currently in use.

This solution consists in evaluating the size of the payload to be transferred and proceeding with the described method if the size is suitable (e.g. greater than 20 bytes). The 8 bits from bit 62 to bit 69 of the UDP message header can be used to state the compact nature of the UDP message, e.g. by setting one of the bits to 1 (this bits are currently not used and set to 0 by default).

Specifically, reference 300 in the diagram in figure 16 indicates any step in which the need to send a message susceptible of being transported in a UDP message is generated followed by a step 302 in which the payload is compressed according to the method described above.

A subsequent step 304 consists in generating the UDP message header in the terms mentioned above. A subsequent step indicated by reference 306 corresponds to the creation of the complete UDP message to prepare for IP transmission (implemented in step 308).

Naturally, numerous changes can be implemented to the construction and embodiments of the invention herein envisaged without departing from the scope of the present invention, as defined by the following claims.

## CLAIMS

1. Method for realising a management activity of at least one managed object (B1, ..., BN) by at least one manager object through a communication network (R),  
5 characterised in that it comprises the following steps:

- providing at least one intermediate object (AG) configured to manage said at least one managed object (B1, ..., BN) according to a data set (1102), said management activity being transformed into a set of results (1112),

10 - providing said data set (1100) from said at least one manager object (A) to said intermediate object (AG),

- managing said at least one managed object (B1, ..., Bn) through said at least one intermediate object (AG), to generate said set of results,

15 - transferring (1108) said set of results from said at least one intermediate object (AG) to said at least one manager object (A).

2. Method according to claim 1, characterised in that it comprises the step of establishing the communication  
20 between said at least one manager object (A) and said at least one intermediate object via UDP protocol.

3. Method according to claim 1 or claim 2, characterised in that it comprises the following steps:

25 - managing at least one further managed object (Bk, ..., BN) directly through said at least one manager object (A), and

- managing said at least one managed object (B1, B2, B3) by said at least one manager object (A) via said intermediate object (AG).

30 4. Method according to claim 3, characterised in that it comprises the management of said at least one further managed object (Bk, ..., Bn) and said at least one managed object (B1, B2, B3) through a single communication network (R).

5. Method according to claim 3, characterised in that it comprises the following steps:

- providing a first communication network (RP) for managing said at least one further managed object (B1) directly through said at least one manager object (A) and transferring said data set (1100) and said results set (1118) between said at least one manager object (A) and said at least one further managed object (B1), and
- providing a second communication network (RA) for managing said at least one managed object (B2, B3) through said intermediate object (AG).

6. Method according to any of the previous claims, characterised in that it comprises the steps of providing a plurality of said intermediate objects (AG1, AG2) and managing at least one managed object (B3) through several intermediate objects (AG1, AG2) of said plurality.

7. Method according to any of the previous claims, characterised in that said intermediate object (AG) is provided with respective reception modules (ARX) and transmission modules (ATX) configured so that said at least one manager object (A) sees said intermediate object (AG) essentially as one of said managed objects (B1, ..., Bn).

8. Method according to any of the previous claims, characterised in that said at least one intermediate object (AG) comprises at least one respective management module (MM) configured so that said at least one managed object (B1, ..., BN), which is managed by said at least one intermediate object (AG), sees said at least one intermediate object (AG) essentially as said at least one manager object (A).

9. Method according to any of the previous claims, characterised in that said at least one intermediate object (AG) is provided with one of the following queues:

- an input queue (I) for collecting input messages with

respect to said at least one intermediate object (AG),

- an output queue (U) for collecting output messages from said at least one intermediate object (AG), and

- a working queue (L) for collecting messages inherent  
5 to said management activity performed by said at least one intermediate object (AG) on said at least one managed object (B1, ..., Bn).

10. Method according to claim 9, characterised in that  
it comprises the step of providing, in said at least one  
10 intermediate object (AG), a dedicated module (DC) for analysing the input messages received by said input queue (I).

11. Method according to claim 9 or claim 10,  
characterised in that it comprises the following steps:

15 - providing, in said at least one intermediate object (AG), an activity co-ordinating module (CA) for implementing at least one of the following functions:

- instantiating at least one concurrent process,
- updating activity status of the requests in said  
20 working queue L, and

- creating statistic check messages to be sent to said at least one manager object (A) through said output queue (U).

12. Method according to any of the previous claims,  
25 characterised in that it comprises the step of providing a plurality of protocol management modules (MP1, MP2, MP3) configured to establish communication to said at least one managed object (B1, ..., BN) through respective different protocols in said at least one intermediate object (AG).

30 13. Method according to any of the previous claims, characterised in that it comprises the step of establishing the communication between said at least one manager object (A) and said at least one intermediate object (AG) by subjecting at least one part of the respective messages to

a compression operation (302; 104, 204).

14. Method according to claim 13, characterised in that said compression operation is based on the acknowledgement of a sequence which appears periodically in the message.

5 15. Method according to claim 14, characterised in that said compression operation implements a gzip type method, such as zLib.

16. Method according to claim 2 and any of the claims from 13 to 15, characterised in that it comprises the step  
10 of indicating that compression of the message transferred by UDP is done.

17. Method according to claim 16, characterised in that a bit field in the UDP header is used to indicate that the compression operation (302) is done.

15 18. Method according to claim 17, characterised in that bits comprised in the range from bit 62 to bit 69 in the UDP header are used in indicate that the compression operation (302) is done.

19. Method according to claim 18, characterised in that  
20 comprises the step of setting at least one of the bits from 62 to 69 of the UDP message header to 1.

20. Method according to any of the claims from 13 to 19, characterised in that the communication between said at least one manager object (A) and said at least one  
25 intermediate object (AG) is implemented by means of SNMP messages, and comprises the following steps during the compression step:

- reading (100) the entire SNMP message,
- encoding (102) the read message in hexadecimal  
30 format, and
- subjecting the message encoded in hexadecimal format to compression (104).

21. Method according to any of the claims from 13 to 19, characterised in that communication between said at

least one manager object (A) and said at least one intermediate object (AG) is implemented by means of SNMP messages, comprises the following steps during the reception step:

- 5       - subjecting the received message to decompression (204) complementary to said compression operation, to obtain a message subjected to decoding in hexadecimal format,
- decoding (202) the message from the hexadecimal
- 10       format, and
- reconstructing (200) the entire SNMP message from said decoded message.

22. Method according to claim 20 or claim 21, characterised in that it comprises a nesting operation in a

15       standard SNMP message for the transmission of the message subjected to said compression operation (104).

23. Method according to claim 22, characterised in that it comprises the following steps during transmission:

- reading (108) the message subjected to said
- 20       compression operation (104) in bytes and transposing (110) it into a corresponding ASCII character message,
- generating (112) a variable binding set comprising a first OID indicating the original file size and subsequent OID/value pairs which carry portions of said message
- 25       subjected to said compression operation (104) transposed into ASCII characters,
- reconstructing SNMP message header data,
- encoding (114) the resulting SNMP message in hexadecimal format to generate the UDP payload, and
- 30       - transferring (116) the UDP payload generated in this way.

24. Method according to claim 22 or claim 23, characterised in that it comprises the following steps during reception:

- receiving the message subjected to said compression operation as an UDP payload (216),

- subjecting the payload received in this way to a hexadecimal decoding operation (214),

5       - acknowledging and assembling (212) the variable binding of the message subjected to hexadecimal decoding,

- subjecting the message subjected to said acknowledging and assembling operation (212) to binary ASCII decoding (210), and

10       - subjecting the decoded message in binary form to said decompression operation (204).

25. Method according to claim 20 or claim 21, characterised in that it comprises the step of integrating the message subjected to said compression operation (104) through UDP nesting for the transmission of the message subjected to said compression operation (104).

26. Method according to claim 25, characterised in that it comprises the following steps during transmission:

20       - configuring said message subjected to said compression operation (104) as a Protocol Data Unit (PDU) payload, and

- transferring the payload created in this way to a given receiver port.

27. Method according to claim 25 or claim 26, characterised in that it comprises the following steps during reception:

- receiving said message as a payload of a PDU UDP received at a receiver port, and

- extracting said payload from said PDU.

30       28. Method according to claim 26 or claim 27, characterised in that it comprises the step of transmitting a synchronisation message (1106) of the SNMP type indicating said transmission port and/or said reception port between said at least one manager object (A) and said



at least one intermediate object (AG).

29. System for managing communication networks comprising at least one manager object (A) and at least one managed object (B1, ..., Bn), characterised in that it  
5 comprises at least one intermediate object (AG) implementing the method according to any of the claims from 1 to 28.

30. Software modules which can be directly loaded into the internal memory of at least a computer and comprising  
10 portions of software code to implement the method according to any of the claims from 1 to 28 when the software modules are run by at least one computer.

Fig. 1

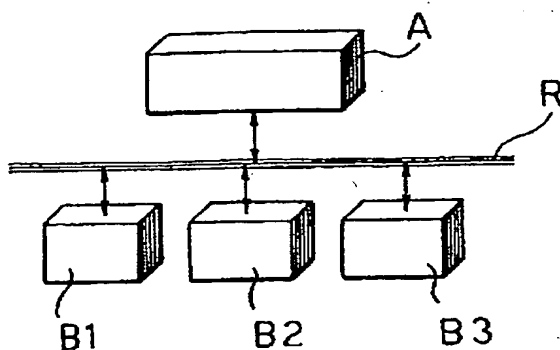


Fig. 2

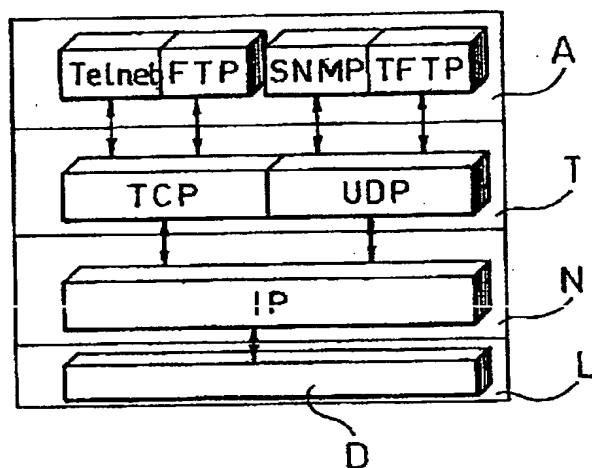
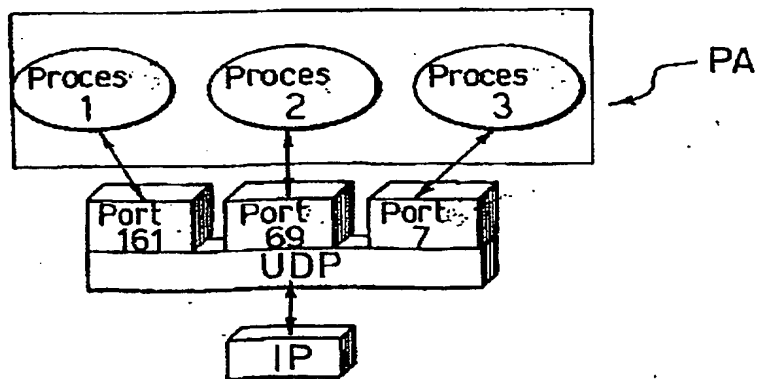


Fig. 3



10/511188

2/7

Fig. 4

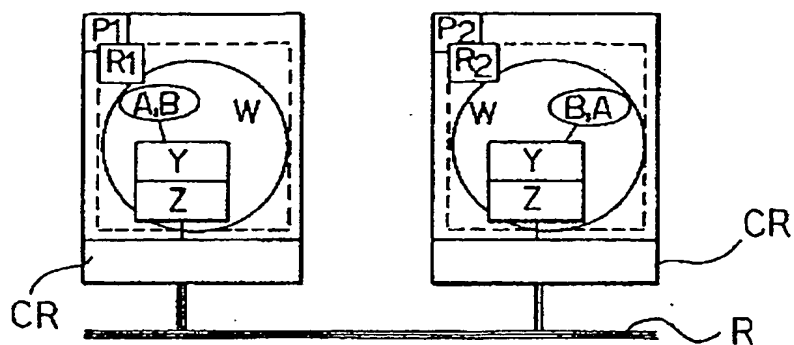


Fig. 5

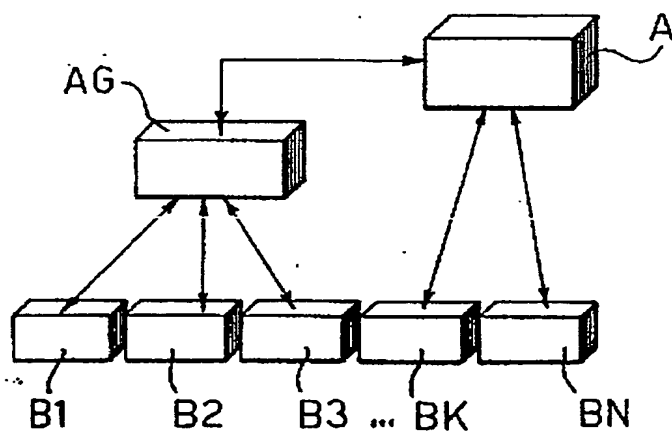
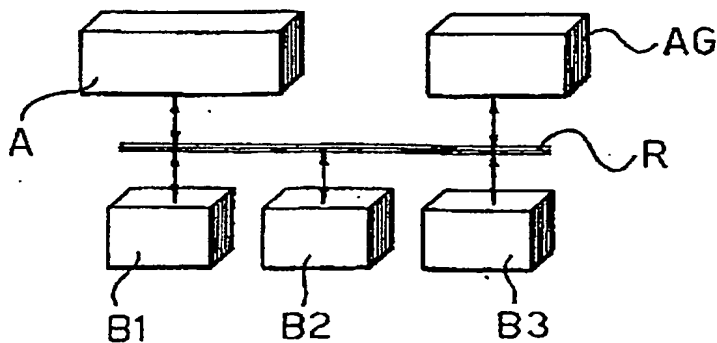


Fig. 6



10/511188

Fig. 7

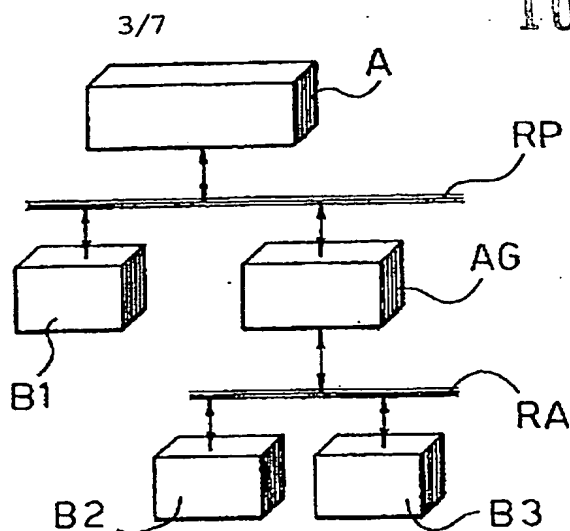


Fig. 8

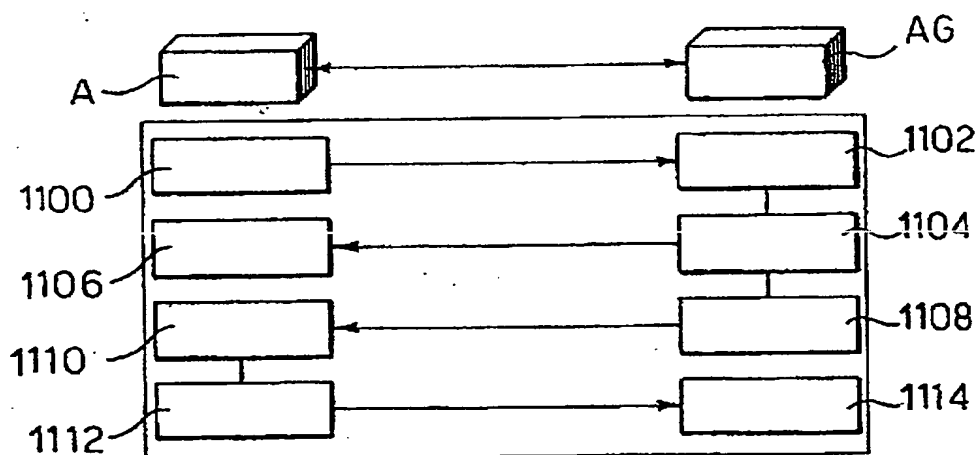


Fig. 9

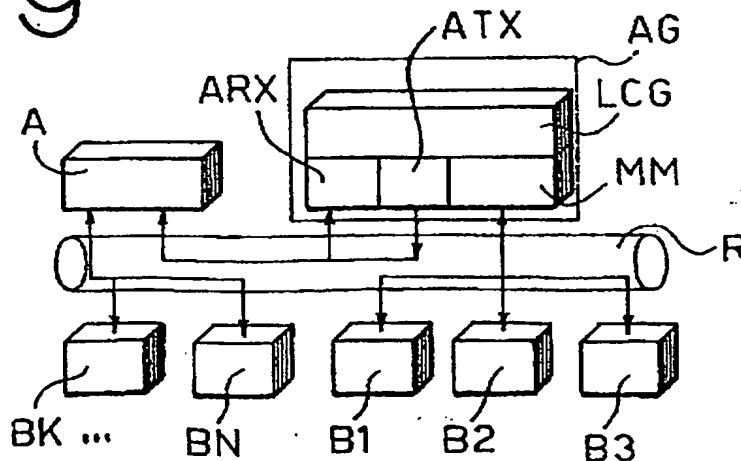


Fig. 10

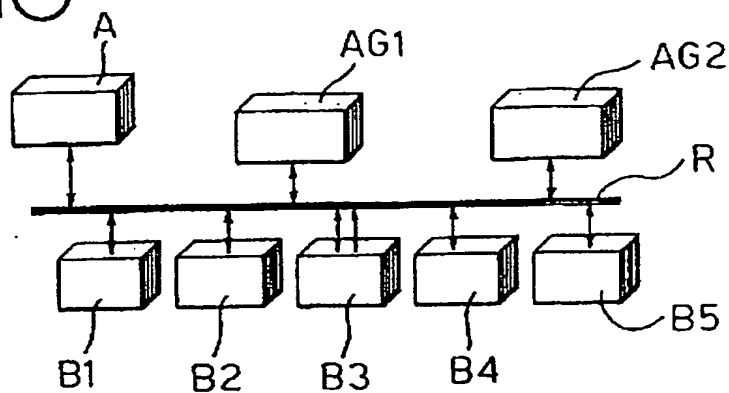


Fig. 11

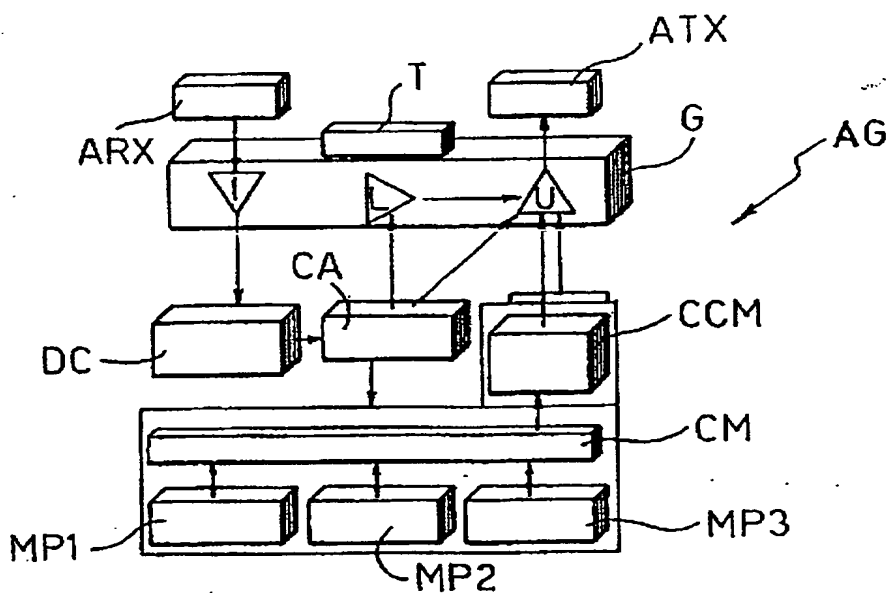


Fig. 12

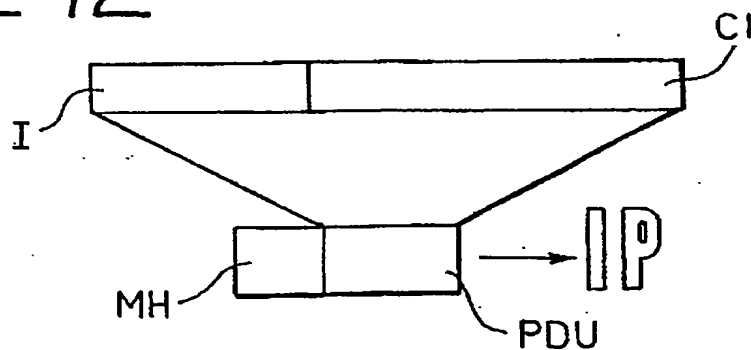


Fig. 13

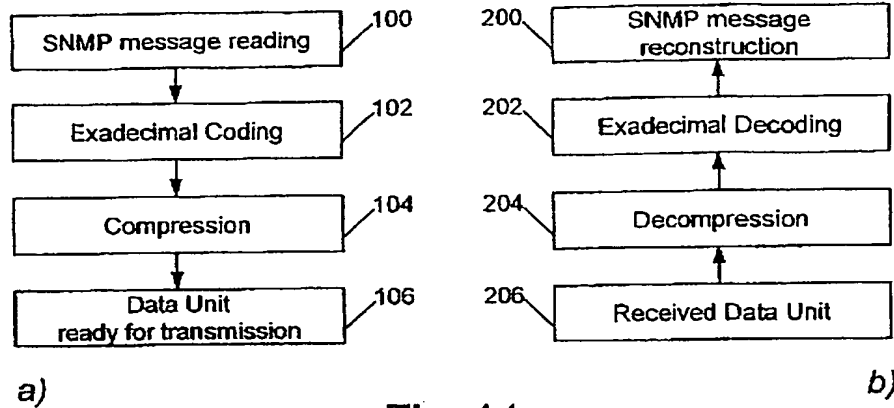


Fig. 14

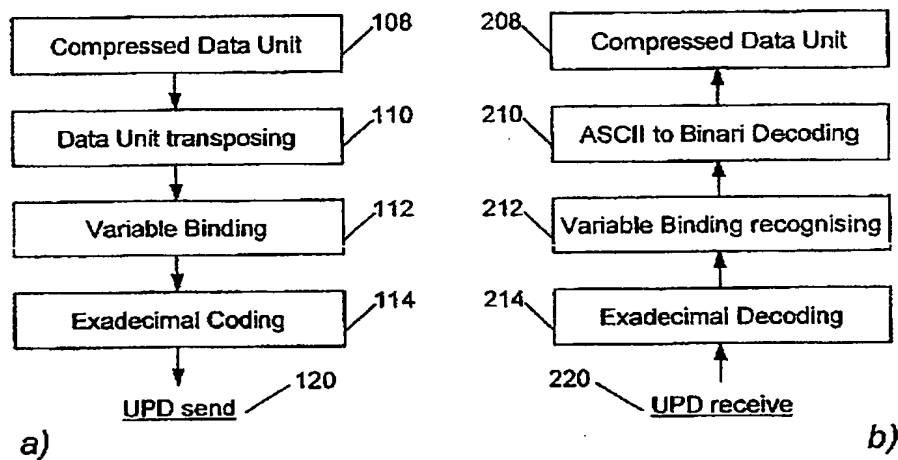
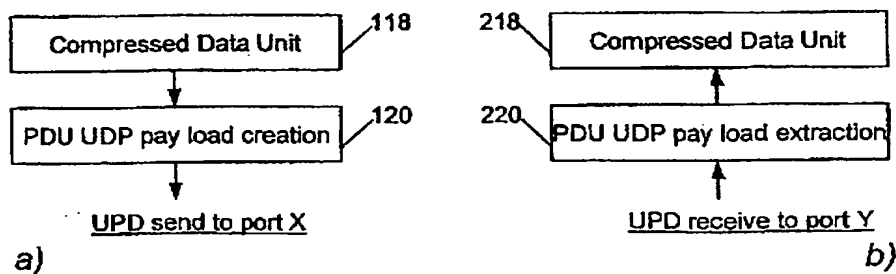


Fig. 15



10/511188

6/7

Fig. 18

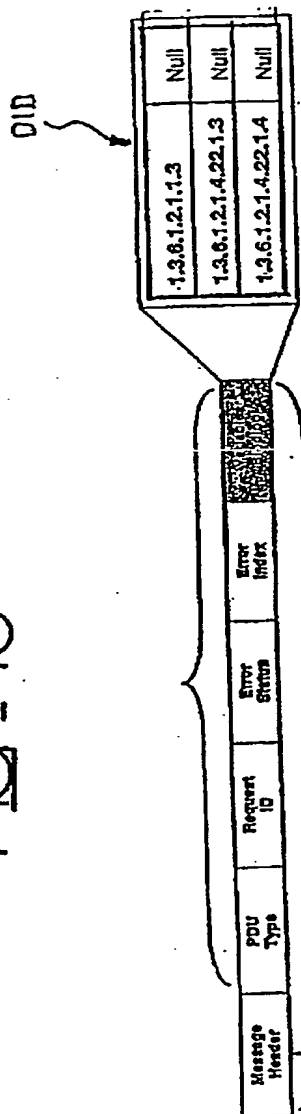
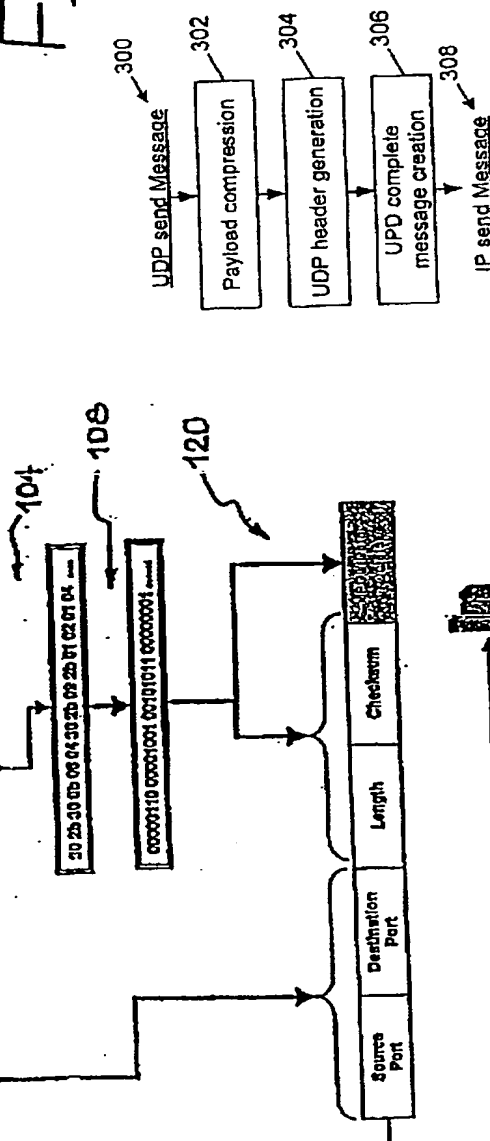
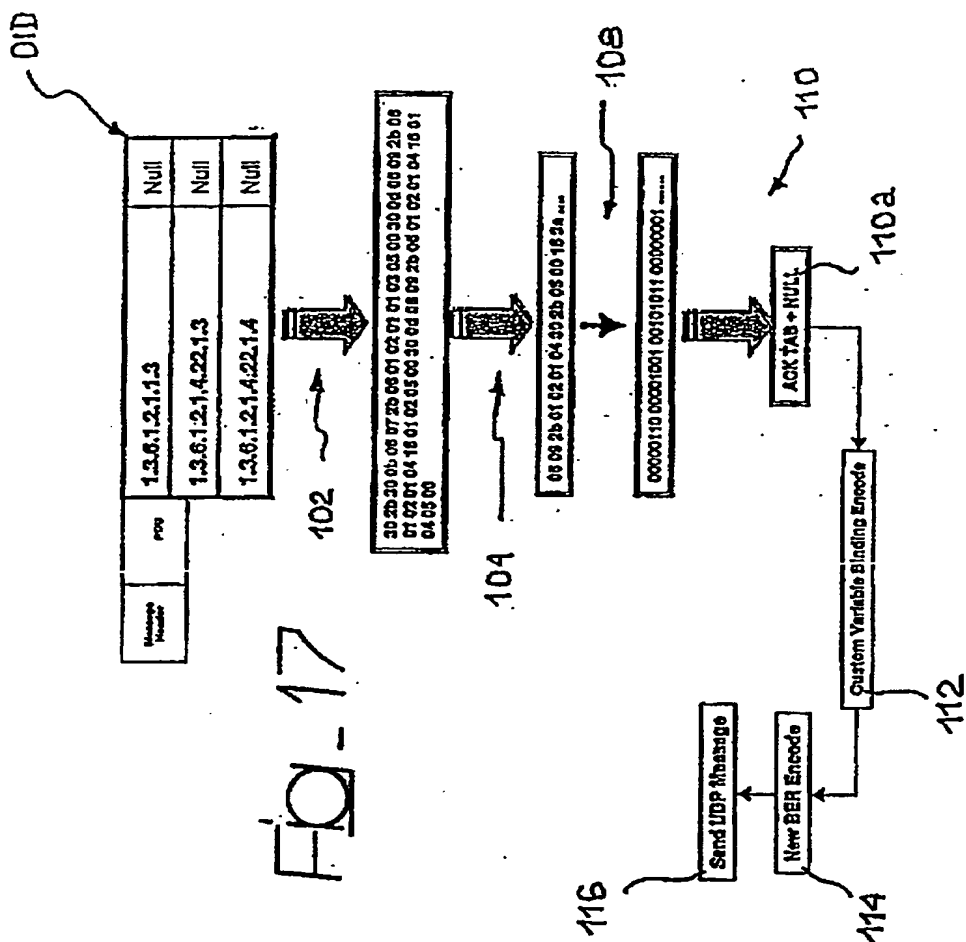


Fig. 16



10/511188

7/7





# INTERNATIONAL SEARCH REPORT

International Application No

PCT/EP 03/03625

## A. CLASSIFICATION OF SUBJECT MATTER

IPC 7 H04L12/24

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 7 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the International search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, INSPEC

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	<p>YANG J ET AL: "A SCALABLE, WEB-BASED ARCHITECTURE FOR HIERARCHICAL NETWORK MANAGEMENT"</p> <p>1999 IEEE GLOBAL TELECOMMUNICATIONS CONFERENCE. GLOBECOM'99. SEAMLESS INTERCONNECTION FOR UNIVERSAL SERVICES. RIO DE JANEIRO, BRAZIL, DEC. 5-9, 1999, IEEE GLOBAL TELECOMMUNICATIONS CONFERENCE, NEW YORK, NY: IEEE, US, vol. 3, 5 December 1999 (1999-12-05), pages 1882-1888, XP001003926</p> <p>ISBN: 0-7803-5797-3</p> <p>abstract</p>	<p>1-5, 12, 29, 30</p>
A	<p>page 1883, right-hand column, paragraph 2</p> <p>-page 1885, right-hand column, paragraph 3</p> <p>page 1887, left-hand column, paragraph 2 - paragraph 3</p> <p>page 1888, left-hand column, paragraph 5</p> <p>-/--</p>	<p>6-11, 13-28</p>

☒ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

### \* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search

19 August 2003

Date of mailing of the international search report

28/08/2003

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2  
NL - 2280 HV Rijswijk  
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,  
Fax: (+31-70) 340-3016

Authorized officer

Bermudez, M.D.

# INTERNATIONAL SEARCH REPORT

Internat. Application No.  
PCT/L. J3/03625

## C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	<p>US 5 651 006 A (NAKAZAKI SHINICHI ET AL)</p> <p>22 July 1997 (1997-07-22)</p> <p>abstract</p> <p>column 2, line 49 -column 3, line 12</p>	1,4,5,7, 8,29,30
A	<p>column 7, line 1 - line 41</p> <p>column 8, line 40 - line 52</p>	2,3,6, 9-28

# INTERNATIONAL SEARCH REPORT

Annex to the international patent application  
Annex on patent family members

International Application No

PCT/E. J3/03625

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5651006 A	22-07-1997	JP 7334445 A	22-12-1995

**This Page is Inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record**

**BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☒ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☒ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: \_\_\_\_\_

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**